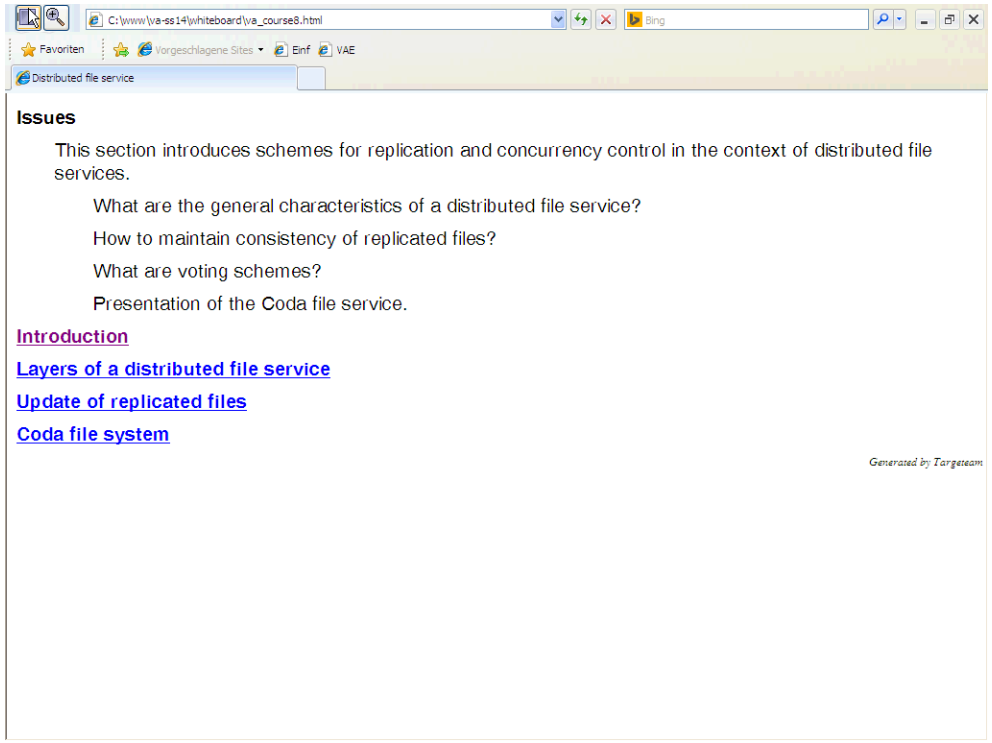
Script **generated by TTT**

Title: Distributed_Applications (01.07.2014)

Date: Tue Jul 01 14:31:36 CEST 2014

Duration: 88:28 min

Pages: 24



Introduction

When a group of programmers has the task to build a distributed application, in addition to distributed code management there is also the need for distributed file services.

Definitions

Motivation for replicated files

Two consistency types

Replica placement

Generated by Targeteam

Layer semantics

Each layer of the distributed file service has a specific task.

Name/directory service

placement of files; file relocation for load balancing and performance improvement; localization of the server which manages the referenced file.

mapping of textual file names to file references (server name and file identifier).

Replication service

file replication for shorter response times and increased availability.

handles data consistency and the multiple copy update problem.

Transaction service

provides a mechanism for grouping of elementary operations so as to execute them atomically;

mechanisms for concurrency control;

Mechanisms for reboot after errors;

File service

relates file identifiers to particular files;

performs read and write operations on the file content and file attributes.

Block service

accesses and allocates disk blocks for the file.

Generated by Targeteam

Distributed file service

Issues

This section introduces schemes for replication and concurrency control in the context of distributed file services.

What are the general characteristics of a distributed file service?

How to maintain consistency of replicated files?

What are voting schemes?

Presentation of the Coda file service.

Introduction

Layers of a distributed file service

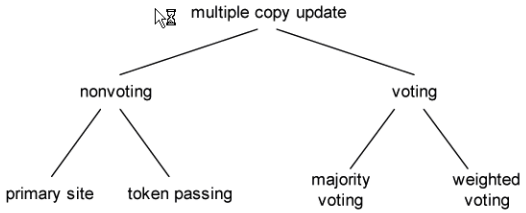
Update of replicated files

Coda file system

Pessimistic concurrency control

Pessimistic concurrency control for data-critical applications, e.g. banking applications. Always access to consistent data.

Classification of pessimistic concurrency control



Primary site

A well-defined file copy, the primary site, serializes and synchronizes all (write) operations.

Token passing

Access to the replicated file (i.e. a file copy) is only permitted, if the client has the token.

Voting schemes

The result of the negotiation between all file replicas determines whether a file access is granted or not.

global consent is necessary, but control is decentralized.

in case of consent, the relevant file block is locked.

Examples: Majority consensus, weighted voting.

Voting schemes

Voting schemes provide pessimistic concurrency control.

Introduction

Voting schemes are algorithms for maintaining mutual consistency of replicates even in situations of computer crashes and network partitionings.

Let us assume, there exist REP replicas of file d.

Let sg(r) be the weight of the vote of computer r; K be the set of all computers considered.

Let the sum of all weights be SUM = $\sum_{r \in K} sg(r)$.

Definitions

Multiple-reader-single-writer strategy

Voting scheme variants

Definitions

Definition: The **votum** for a desired access of a file is defined

as the sum of votes from the set of computers that have voted for the desired access.

Definition: The obtained votum is called successful if the sum of votes from the set of computers that have voted for the desired access is equal to or greater than a lower bound, the so-called **quorum** .

File access is permitted (positive votum), if the following holds

for read access: at least R positive votes (read quorum).

for write access: at least W positive votes (write quorum).

Multiple-reader-single-writer strategy

quorum must support the multiple-reader-single-writer strategy; at least one computer must have the most up-to-date physical replica of the file.

Multiple-reader-single-writer strategy maintains file consistency:

R + W > SUM, i.e. a reader excludes a writer, and vice versa.

W + W > SUM, i.e. only one writer gets a positive vote.

Voting schemes

Voting schemes provide pessimistic concurrency control.

Introduction

Voting schemes are algorithms for maintaining mutual consistency of replicates even in situations of computer crashes and network partitionings.

Let us assume, there exist REP replicas of file d.

Let sg(r) be the weight of the vote of computer r; K be the set of all computers considered.

Let the sum of all weights be SUM = $\sum_{r \in K} sg(r)$.

Definitions

Multiple-reader-single-writer strategy

Voting scheme variants

Majority consensus

In the majority consensus scheme, a votum is successful if at least a majority of computers with a right to vote have voted for the desired access.

the vote of each computer with a replicate has the same weight ("one person one vote"), i.e. $\forall$ r $\in$ K: sg(r) = 1.

A votum is successful if the majority of all relevant computers agree with respect to the desired access:

W = R = REP/2 + 1, if REP is even.

W = R = (REP+1)/2 , if REP is odd.

Voting scheme variants

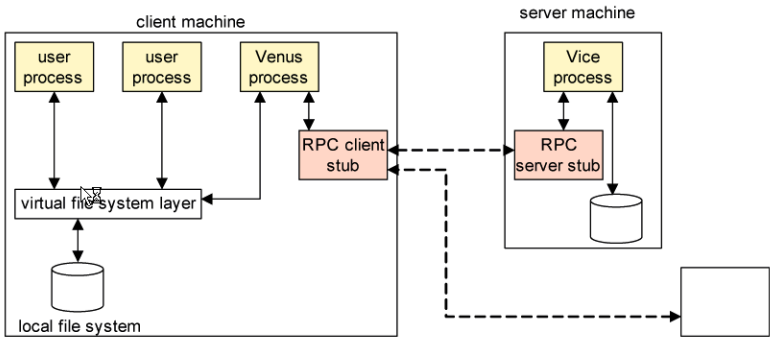
For further variants and details see the book Borghoff/Schlichter, Springer-Verlag, 2000.

Write-All-Read-Any

Majority consensus

Weighted voting

Architecture



Venus processes provide access to files maintained by the Vice file servers.

role is similar to that of an NFS client.

responsible for allowing the client to continue operation even if access to the file servers is (temporarily) impossible.

Naming

Each file is contained in exactly one volume. Distinction between

physical volumes.

logical volume (represents all replicas of a volume).

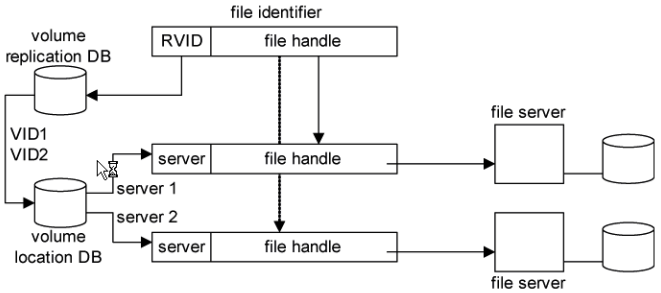
RVID (Replicated Volume Identifier): identifier of a logical volume.

VID (Volume Identifier): identifier of a physical volume.

File identifier

File identifier

Coda assigns each file a 96-bit file identifier.



Naming

Each file is contained in exactly one volume. Distinction between

physical volumes.

logical volume (represents all replicas of a volume).

RVID (Replicated Volume Identifier): identifier of a logical volume.

VID (Volume Identifier): identifier of a physical volume.

File identifier

Generated by Targeteam

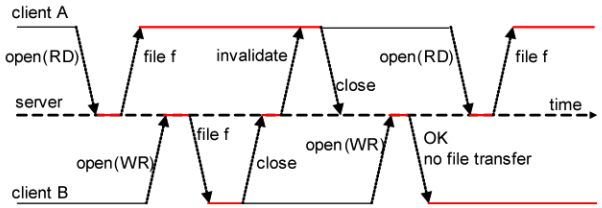
Client caching

When a file is opened, an entire copy of the file is transferred to the client; caching of the file.

client becomes less dependent on the availability of the server.

Cache coherence is maintained by means of callbacks.

- Server records a ***callback promise*** for a client.
- update of the file by a client ⇒ notification to the server ⇒ invalidation message to other clients.



Replication strategy

Coda relies on replication to achieve high availability. It distinguishes between two types of replication.

Client caching

Server replication

Server replication

Coda allows file server to be replicated; the unit of replication is a volume.

Volume Storage Group (VSG): collection of servers that have a copy of a volume.

client's Accessible Volume Storage Group (AVSG): list of those servers in the volume's VSG that the client can contact.

AVSG = {}: client is disconnected.

Coda uses a variant of the "read-one, write-all" update protocol.

Coda version vector

Distributed Applications - Verteilte Anwendungen

- Prof. J. Schlichter
  - Lehrstuhl für Angewandte Informatik / Kooperative Systeme, Fakultät für Informatik, TU München
  - Boltzmannstr. 3, 85748 Garching

    Email: schlichter@in.tum.de

    Tel.: 089-289 18654

    URL: http://www11.in.tum.de/

Overview

Introduction

Architecture of distributed systems

Remote Invocation (RPC/RMI)

Basic mechanisms for distributed applications

Web Services

Design of distributed applications

Distributed file service

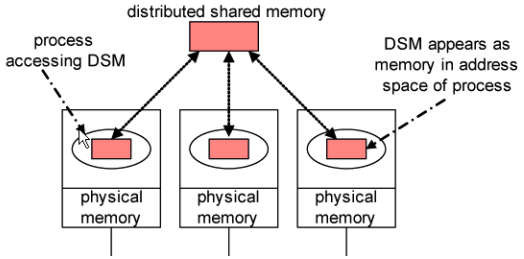
Distributed Shared Memory

Object-based Distributed Systems

Summary

Introduction

Distributed shared memory (DSM) is an abstraction used for sharing data between computers that do not share physical memory.



Programming model

Message passing model

variables have to be marshalled from one process, transmitted and unmarshalled into other variables at the receiving process.

Distributed shared memory

the involved processes access the shared variables directly; no marshalling necessary.

processes may communicate via DSM even if they have non-overlapping lifetimes.

Implementation approaches

in hardware

shared memory multiprocessor architectures, e.g. NUMA architecture.

in middleware

language support such as Linda tuple space or JavaSpaces.

Distributed Shared Memory

Issues of the section

implicit communication via shared memory

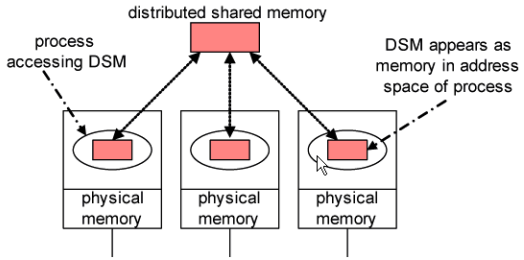
what is the Linda tuple space?

Javaspaces as modern tuple space

Introduction
Programming model
Consistency model
Tuple space
Object Space

Introduction

Distributed shared memory (DSM) is an abstraction used for sharing data between computers that do not share physical memory.



The content of DSM may be replicated by caching it at the separate computers;

data is read from the local replica.

updates have to be propagated to the other replicas of the shared memory.

Approaches to keep the replicas consistent

Write-update

updates are made locally and multicast to all replicas possessing a copy of the data item.

the remote data items are modified immediately.

Write-invalidate

before an update takes place, a multicast message is sent to all copies to invalidate them;

acknowledgement by the remote sites before the write can take place.

other processes are prevented to access the blocked data item.

the update is propagated to all copies, and the blocking is removed.